EvAg: A Scalable Peer-to-Peer Evolutionary Algorithm

Laredo, J.L.J.; Eiben, A.E.; van Steen, M.R.; Merelo, J.J.

published in
Genetic programming and evolvable machines
2010

DOI (link to publisher)
10.1007/s10710-009-9096-z

document version
Publisher's PDF, also known as Version of record

Genet Program Evolvable Mach (2010) 11:227–246
DOI 10.1007/s10710-009-9096-z

ORIGINAL PAPER

EvAg: a scalable peer-to-peer evolutionary algorithm

J. L. J. Laredo · A. E. Eiben · M. van Steen · J. J. Merelo

Received: 15 May 2009 / Revised: 29 September 2009 / Published online: 25 November 2009

Abstract This paper studies the scalability of an Evolutionary Algorithm (EA) whose population is structured by means of a gossiping protocol and where the evolutionary operators act exclusively within the local neighborhoods. This makes the algorithm inherently suited for parallel execution in a peer-to-peer fashion which, in turn, offers great advantages when dealing with computationally expensive problems because distributed execution implies massive scalability. In this paper we show another advantage of this algorithm: We experimentally demonstrate that it scales up better than traditional alternatives even when executed in a sequential fashion. In particular, we analyze the behavior of several EAs on well-known deceptive trap functions with varying sizes and levels of deceptiveness. The results show that the new EA requires smaller optimal population sizes and fewer fitness evaluations to reach solutions. The relative advantage of the new EA is more outstanding as problem hardness and size increase. In some cases the new algorithm reduces the computational efforts of the traditional EAs by several orders of magnitude.

Keywords Peer-to-peer computing · Evolutionary algorithms · Scalability analysis · Diversity

J. L. J. Laredo (✉) · J. J. Merelo
University of Granada, ATC-ETSIT, C. Periodista Daniel Saucedo Aranda, 18071 Granada, Spain
e-mail: juanlu@geneura.ugr.es

J. J. Merelo
e-mail: jmerelo@geneura.ugr.es

A. E. Eiben · M. van Steen
Department of Computer Science, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
e-mail: gusz@cs.vu.nl

M. van Steen
e-mail: steen@cs.vu.nl

1 Introduction

Evolutionary Algorithms (EAs) are a set of population-based stochastic search techniques able to solve optimization problems in reasonable time. However, for very demanding applications and large problem instances the computational efforts (running times) of EAs can be high. Fortunately, EAs are rather easy to execute in a parallel fashion offering a straightforward way to improving scale-up properties [30]. A remaining challenge in parallel EAs is the central management of the evolutionary cycle (parent selection, reproduction, survivor selection) imposing limitations on scalability.

Within this context, Peer-to-Peer (P2P) systems provide a powerful parallel infrastructure able to constitute a single virtual computer composed of a potentially large number of interconnected resources without central control [33]. Such a computing paradigm defines a rich set of topologies for the interconnection of nodes at application level, so-called overlay networks. The main idea behind a P2P EA is to designate each individual in the population as a peer and adopt a population structure defined by a P2P overlay network [39]. Then any given individual has a limited number of neighbors and mating is restricted to the P2P neighborhood.

Interactions in such a spatially structured EA can be visualized as a graph where vertices represent individuals and edges the relationships between them [35]. In this sense, a traditional unstructured population (a.k.a. panmictic population) is represented as a complete graph, whereas other approaches define a richer set of population structures such as regular lattices [13], toroid [11] or small-world [12, 14, 31]. Based on these studies, the Evolvable Agent model (EvAg) presented in [22] is a distributed and decentralized P2P EA in which the population structure is defined by a gossiping protocol called newscast that behaves asymptotically as a small-world graph [19, 20].

This paper analyzes the scalability of such an approach from two perspectives. First, we investigate how population sizes scale with increasing problem size and difficulty. To this end, we follow the population sizing theory [15] which states that there is an optimal population size for a given problem instance that can be determined under some conditions. In particular, we use the bisection method [32] that determines the minimum population size N for a selectorecombinative Genetic Algorithm (GA), i.e., a GA without mutation. The assumption of a selectorecombinative GA is commonly made in population sizing studies as the only source of diversity is then the initial population which stands for a worst case analysis [28]. Second, we look at the scale-up properties of the average number of evaluations to a solution (AES), which is a device independent measure of computational effort that can be used for all EAs, with or without mutation [8].

In order to assess EvAg scalability, we conduct experiments on trap functions with different levels of deceptiveness [1]. These functions represent a set of decomposable problems based on unitation and composed of m sub-functions in which the total fitness is additively calculated by summing the partial fitness of every sub-function. Hence, it is easy to scale the problem size by varying the number of sub-functions m. In addition, each sub-function is composed of k bits representing the building block (BB) size, and only one of the 2^k possible

combinations belongs to the optimal solution. By varying the value of k we can vary the level of problem difficulty, from non-deceptive, through quasi-deceptive to deceptive problems [7].

The rest of this paper is structured as follows. Section 2 reviews the state of the art literature related to P2P EAs. Section 3 outlines the overall architecture of the Evolvable Agent model and provides some insights into the role that the population structure plays on the EA performance. Section 4 explains the followed methodology and experiments, including a description of trap functions and the tuning of the population size. Section 5 presents the results of the experiments and analyzes the scalability of the EAs tested. Finally, some conclusions are drawn in Sect. 6 and some future lines of work proposed.

2 Related work

According to [2] parallel EA implementation is approached mainly by means of three methodologies: master-slave, islands and fine grained models. However, not all the models can be easily adapted to P2P systems due to issues such as decentralization, massive scalability or fault tolerance.

- In the *master-slave mode* the algorithm runs on the master node and the individuals are sent for evaluation to the slaves, in an approach usually called *farming*. Such an architecture does not match decentralized structures and has the obvious drawback of the master being a single point of failure. In addition, scalability is often limited by master performance and incoming bandwidth.
- One of the most usual and widely studied approaches in parallel EAs is the *Island model* (see [6] for a survey). The idea behind this model is that the global panmictic population is split in several sub-populations or demes called islands. The communication between islands is defined by a given topology, through which they exchange individuals (migrants) with a certain rate and frequency. The migration follows a selection policy in the source island and a replacement policy in the target one. Practitioners generally use a fixed population size P in studies of scalability, a fixed number of islands N and a population size per island of P/n where $n = 1, \ldots, N$. That is the case described in [18] where the authors show how the algorithmic results are highly sensitive to the number of islands making the model vulnerable for highly dynamic systems such as P2P.
- In *fine-grained approaches* every individual within the population is placed on its own processor and most of the works focus on the algorithmic effects of using different neighborhood policies. For example, [13] studies the impact of regular lattices on the selection pressure, of different graph structures such a toroid in [11] or small-world in [12]. This last population structure has been empirically shown to be competitive against panmictic EAs in [14, 31]. Finally, under the idea of individuals evolving within a given set of neighbors, [39] presents a fine-grained P2P EA showing the suitability of the approach for decentralized systems.

A first insight from Alba and Tomassini's paper is that P2P EAs models should be closer to fine grained approaches in order to be scalable. However,

studying scalability in P2P optimization is not straightforward and is usually approached in two complementary ways, using real environments [25] or simulations [4]; either way presents its own advantages and drawbacks. On the one hand, performing a real massively distributed and decentralized experiment presents some challenges that, so far, pose a whole set of practical problems beyond the state of the art. The main reason is the difficulty to gather a large amount of reliable resources. Whenever the study of scalability is reduced to a few peers, no conclusions about massive scalability can be drawn. However, if the amount of peers is large enough, other questions, such as fault tolerance, arise [29]. Some of the most relevant works that have tackled scalability are detailed bellow:

- The *DREAM* project, [3], is one of the pioneering frameworks for P2P Evolutionary Computation (EC). The project focuses on the distributed processing of EAs and uses the P2P engine DRM (Distributed Resource Machine) which is an implementation of the newscast protocol [19]. However, the island-based parallelization of *DREAM* was shown in [21] to be insufficient for tackling large-scale decentralized scenarios. There are some other frameworks based on *DREAM* as *G2DGA* [5] which uses G2P2P instead of DRM and focuses on Genetic Algorithms among all the EAs paradigms.
- Lee [27] proposes a parallel system for EC using the P2P framework JADE. The optimization is performed by three kinds of agents: state agents for controlling whether the peer is active or not, mobile agents for performing the evolutionary computation and the synchronizing agent, a centralized agent for synchronizing all the active peers. The algorithm's execution time speeds-up linearly but the scalability analysis is limited to eight nodes from which no conclusions about true scalability can be drawn.
- Folino and Spezzano [10] propose the *P-CAGE* environment for EC in P2P systems which is a hybrid model combining islands with cellular EAs. Every peer holds an island and every island a cellular EA. Despite results outperforming canonical EAs (either regarding execution time or convergence speed), the scalability analysis is limited to ten peers and the algorithm yields the best performance with five peers which points to poor scalability.

On the other hand, using simulations simplifies the analysis and allows focusing on the structural design since restrictions like the harnessing of computing power or the peers' failures disappear. The drawback in this case is that simulations imply a certain number of assumptions about the real environment; hence, they have to be well stated (e.g., representing a pessimistic scenario as in [4]). In addition to approaches for real environments, some other works in the literature face the design of P2P EAs by means of simulations, focusing on the viability of the approaches rather than dealing with the harnessing of computing power.

- The self-organizing topology evolutionary algorithm (*SOTEA*) in [38] is an EA designed for the sake of diversity maintenance. To this end, the authors focus on a self-organized population structure with the shape of a complex network. The

network co-evolves with the EA by following two rules (from which a power law population structure emerges):

1. Reproduction rule: When a new offspring is created, SOTEA adds a new node, this node is linked to its parent (asexual reproduction). The parent's connections are inherited by the offspring with certain probability P_{add}. In addition, all inherited connections are lost by the parent with probability P_{remove}.
2. Competition rule: A randomly selected individual competes with its least fit neighbor. From such a competition, the loser is killed and the winner inherits all its connections. By following these two rules, SOTEA keeps a better population diversity than the Cellular and Panmictic GA used as a baseline for comparison.

– Within the line of self-organized algorithms, [24] and, [39] present a P2P EA with two particular properties: autonomous selection and natural reproduction. Autonomous selection means that the individuals in the population decide on their own whether and when they want to reproduce and to survive without any central control. To this end, they use information on their own fitness and estimations about the total population to support decision making. The second special feature, natural reproduction, means that birth and death decoupled [9]. That is, an individual can be removed without being replaced by a child and a child can be born without removing an existing individual first. This is highly uncommon in EAs and as a consequence, the population size varies at run-time (just like in natural evolution) and a self-adjusting selection pressure mechanism is needed to prevent population implosions or explosions.

Following the same idea of decentralization, we presented in [22] a formal model for P2P EAs, it is the *Evolvable Agent model* that we analyze in this paper for large-scale scenarios.

3 Overall model description

The overall architecture of our spatially structured EA consists of a population of Evolvable Agents (EvAg), described below and initially proposed in [22], whose design objective is to carry out the main steps of evolutionary computation (selection, variation and evaluation of individuals [8]). To this aim, each EvAg evolves within its neighborhood which is locally maintained by the P2P protocol newscast. Newscast runs on every node and defines a self-organizing graph that dynamically maintains constant some graphs properties such as a low average path length or a high clustering coefficient from which emerges a small-world behavior.

3.1 Evolvable agent

We deliberately leave the definition for agent open with the basic constraint of being just an encapsulated processing unit. This way future works could easily extend the

Table 1 EvAg

```
S_actual ⇐ Initialize Agent
loop
  Sols ⇐ Local Selection(Neighbors_EvAg)
  S_new ⇐ Recombination(Sols,P_c)
  S_new' ⇐ Mutation(S_new,P_m)
  Evaluate(S_new')
  if S_new' better than S_actual then
    S_actual ⇐ S_new'
  end if
end loop
```

EvAg definition (i.e., behavioral learning between agents, self-adaptive population size adjustment during run-time [39] or load balancing mechanisms within a real network [3]).

Table 1 shows the pseudo-code of an EvAg where the agent owns an evolving solution (S_{actual}).

The mate selection takes place locally within a given neighborhood where each agent selects the current solution from other agents (S_{actual}). Selected solutions are stored in *Sols* ready to be recombined and mutated. Within this process a new solution $S_{\text{new}'}$ is generated. If the newly generated solution $S_{\text{new}'}$ is better than the old one S_{actual}, it replaces the current solution.

3.2 Population structure as a complex network

To help understand the role of the population structure in a P2P EA, this section introduces the structural design of a simple and easy understandable complex network proposed in [37]. As described by the authors, the procedure for building a small-world graph can start from a ring lattice with n vertices and k edges per vertex. With a given probability p, each edge is rewired at random. Since the procedure does not allow duplicate edges, no edge is generated whenever it matches an existing one. This way for a rewiring factor of $p = 0$ the ring lattice is kept while for $p = 1$ a random graph is generated. It has been shown that already for small values of p, the average distance between two nodes decreases rapidly.

Figure 1 shows three instances of the Watts–Strogatz model in which the small-world graph preserves the high clustering coefficient of regular lattices and the small average path length of random graphs. Despite having a larger average path length than complete graphs, the inhomogeneity in such kind of topologies was shown in [12] to induce qualitatively similar selection pressures on EAs compared to panmictic population structures.

The influence in the environmental selection pressure of such population structures can be represented by their takeover time curves. Goldberg and Deb [16] defines the takeover time as the time that it takes for a single, best individual to take over the entire population without any other mechanism than selection. Hence, takeover time is the proportion of best individuals formulated as a function of time.

Figure 2 shows that the takeover time curve in the Watts–Strogatz population structure is similar to a panmictic one meaning that the induced selection pressures with both topologies are roughly equivalent. As in Watts–Strogatz small-world

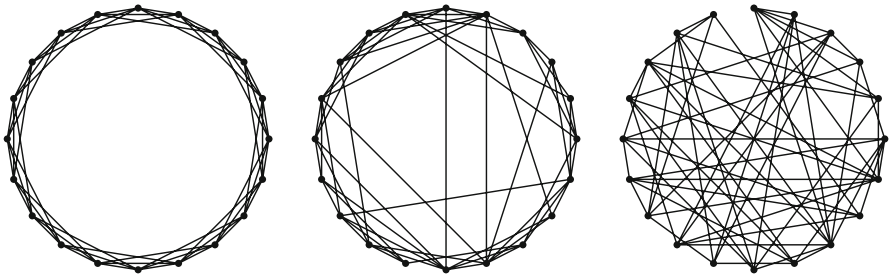

Fig. 1 Watts-Strogatz graphs with $n = 20$ and $k = 6$. From *left* to *right*, the original ring lattice for $p = 0$, a small-world graph for $p = 0.2$ and a random graph for $p = 1$

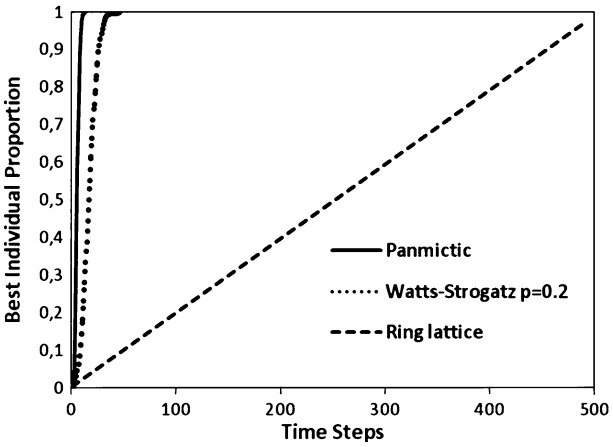

Fig. 2 Takeover time curves in a panmictic population structure, Watts–Strogatz population structure with $p = 0.2$ and the original ring lattice with $k = 2$. Results are averaged from 50 independent runs for a population size of $n = 1,600$ and binary tournament

topologies, this paper shows that P2P topologies can induce similar selection pressures to the panmictic one, allowing in addition a better scalability behavior at the lower edge cardinality of P2P systems.

3.3 Newscast as population structure

In principle, our method places no restrictions on the choice of population structure, but this choice will have an impact on the dynamics of the algorithm. In this paper, the newscast protocol is considered as neighborhood policy and topology builder.

Newscast is a dynamic and self-organized gossiping protocol for the maintenance of unstructured P2P overlay networks [19] that we use as population structure for our EA. Table 2 shows the pseudo-code of the main tasks involved in the self-organized maintenance of a newscast topology. Each node keeps its own set of neighbors by a cache that contains $c \in \mathbb{N}$ entries, referring to c other nodes in the population without duplicates. Each entry provides the following information about

Table 2 Newscast protocol in node EvAg_i

Active Mode
loop
  $EvAg_j \Leftarrow$ Uniformly random selected node from $Cache_i$
  send $Cache_i$ to $EvAg_j$
  receive $Cache_j$ from $EvAg_j$
  $Cache_i \Leftarrow$ Aggregate ($Cache_i$,$Cache_j$)
end loop

Passive Mode
loop
  wait $Cache_j$ from $EvAg_j$
  send $Cache_i$ to $EvAg_j$
  $Cache_i \Leftarrow$ Aggregate ($Cache_i$,$Cache_j$)
end loop

Local Selection($Neighbors_i$)
$[EvAg_h, EvAg_k] \Leftarrow$ Uniformly random selected nodes from $Cache_i$

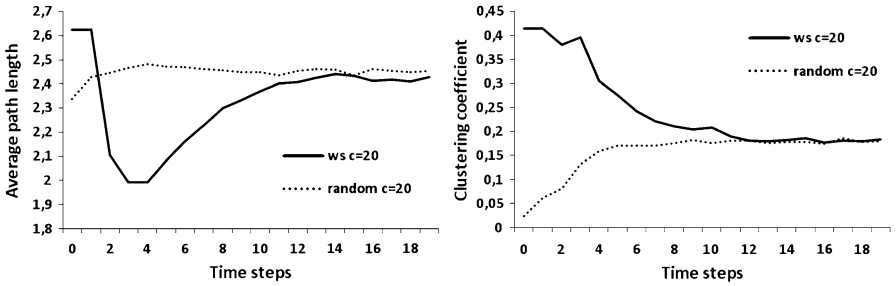

Fig. 3 Convergence of the average path length (*left*) and clustering coefficient (*right*) bootstrapping from a random and a Watts–Strogatz (ws) graph for a number of nodes $n = 1,600$. It can be seen how the algorithm converges to the same values after few cycles showing the independence of the newscast protocol with respect to the initialization criterion

a foreign node: A reference to the node, time-stamp of the entry creation (allowing the replacement of old items), an agent identifier and specific application data.

There are two different tasks that the algorithm carries out within each node. The active mode which pro-actively initiates a cache exchange once every fitness evaluation (a.k.a. cycle) and the passive mode that waits for data-exchange requests. In addition, the local selection procedure provides the EvAg with other agents' current solutions (S_{actual}).

Every cycle each EvAg_i initiates a cache exchange. It selects randomly a neighbor EvAg_j from its $Cache_i$ with uniform probability. Then EvAg_i and EvAg_j exchange their caches and merge them following an aggregation function. In our case, the aggregation consists of picking the freshest c items from $Cache_i \cup Cache_j$ and merging them into a single cache that EvAg_i and EvAg_j will share. Figure 3 shows how the graph properties are dynamically self-organized from such a decentralized process.

The cache size (c) plays here an important role. It represents the maximum degree of a node, and therefore, influences the average path length and the clustering coefficient. For example, Fig. 4 depicts the influence of the cache size on

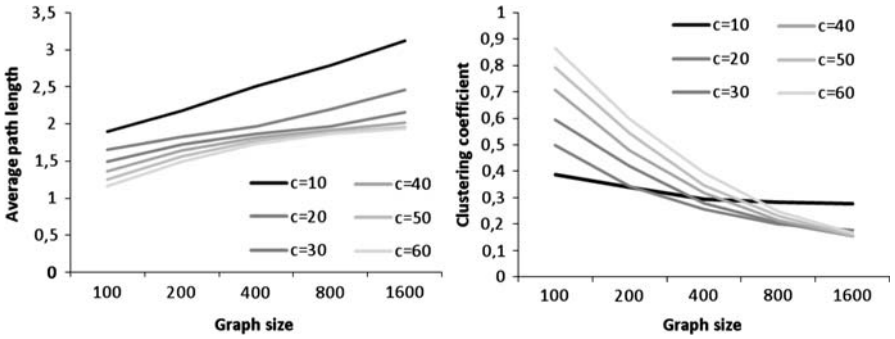

Fig. 4 Average path length (*left*) and clustering coefficient (*right*) for different network and cache sizes (c) averaged from 50 independent runs

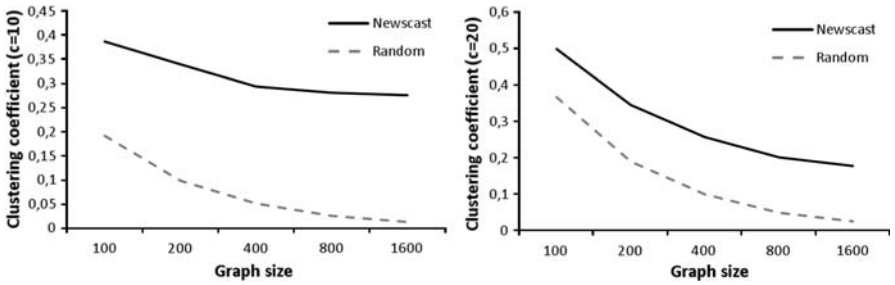

Fig. 5 Clustering coefficients for equivalent random and newscast graphs (i.e., nodes have the same number of edges, ten on the *left* and twenty on the *right*). The higher values in the newscast graphs point to a small-world topology

the average path length and the clustering coefficient for different network sizes. A smaller c implies a higher clustering coefficient and also a higher average path length.

Based on such features, a newscast graph can be classified as small-world; newscast has the small average path length of random graphs but, as shown in Fig. 5, a much higher clustering coefficient [36].

Finally, Fig. 6 shows the takeover time curves for a panmictic and two different parameterized newscast population structures. As explained in the previous section, similar curves denote equivalent selection pressures induced by both kind of topologies. Nevertheless, the node degree in complete graphs is $n - 1$ while the average degree in newscast is approximately $2c$ pointing out a better scalability of the small-world approach given that $c \ll n$. In fact, we use $c = 20$ within all the settings of the experiments. Such value takes into account the recommendations in [19] stating that the intended normal setting of newscast is $c \ll n$ and demonstrating that values from $c = 20$ prevent the spontaneous partitioning of the graph even when it becomes very large. In addition, our work in [26] empirically shows a lack of influence of c on the EvAg performance when $c \in [0.005n, 0.15n]$, where n is the population size.

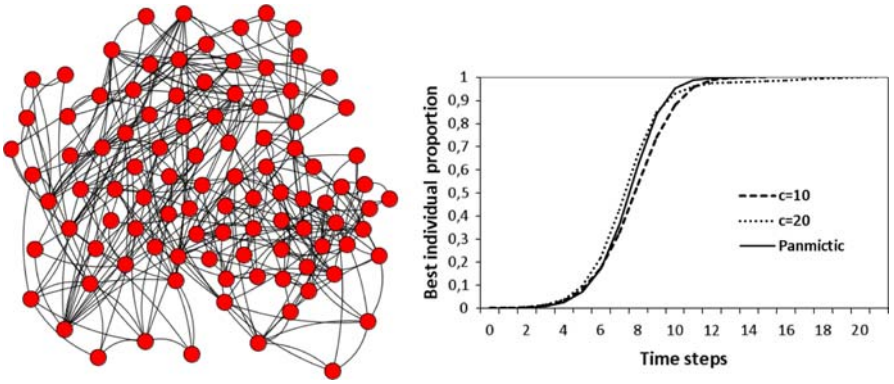

Fig. 6 A snapshot of a newscast population structure on the *left*. It can be visualized that the node degree in newscast is smaller than in the panmictic case. On the *right*, takeover time curves for a panmictic and two newscast population structures with $c = 10$ and $c = 20$. Results are averaged from 50 independent runs for a population size of $n = 1, 600$ and binary tournament

4 Methodology and experimental setup

To investigate the scalability of the EvAg model, experiments were conducted on different trap functions and compared against two canonical GAs, a steady-state GA (SSGA) and a generational GA (GGA). EvAg was simulated following the same steady-state scheme than the SSGA, however, the approaches differ in the population structure. Whereas the population structure of the EvAg model is defined by the newscast protocol, the canonical GAs are panmictic. To analyze scalability three series of experiments were conducted:

In the first series, experiments use selectorecombinative versions of the GAs to estimate optimal population sizes for the different problem instances. The reason for using selectorecombinative GAs is that there are well defined models to establish the population size and the number of evaluations required to solve a given trap function instance [17], meanwhile, to the best of our knowledge there are no such models when using mutation. To this end, in Sect. 4.1 we describe a method for estimating the population size. The underlying idea is that without mutation, the population size scales with respect to the problem instance since it becomes the only source of diversity. In this context, [34] demonstrates the necessity of larger population sizes when tackling larger problem instances.

To overcome the limitation of a mutationless GA, we switch mutation on in the second series of experiments. The aim here is to gain some insight on the influence of such a new source of diversity on the scalability order.

Finally, the third series focus on the analysis of the different approaches when they are equally parameterized. Specifically, the largest instance of the fully deceptive 4-trap function is considered to analyze the convergence of the fitness and the evolution of genotypic diversity.

Within these series, the following metrics were used to assess the results [8]:

- The success rate (SR) measures the algorithm quality as the proportion in which the algorithm is able to find the problem optimum out of all the runs.
- The average number of evaluations to solution (AES) stands for the number of evaluations that the algorithm spends in those runs that yield success. Since results do not follow a normal distribution, we have chosen the number of evaluations to solution in the third quartile ($AESQ_3$) as a central position value, meaning that a 75% of the runs will stay below such a value.
- The convergence of the best fitness.
- The genotypic entropy is a measure of the population diversity defined on the genotypic distances ($H_g(P)$).

$$H_g(P) = -\sum_{j=1}^{N} g_j \log(g_j) \tag{1}$$

where g_j is the fraction $\frac{n_j}{N}$ of individuals in P having a Hamming distance j to the optimal genotype, and N is the number of different distances.

4.1 A method for estimating the population size

The bisection method [32] estimates the optimal population size N to solve a problem instance, that is, the lowest N for which 98% of the runs find the problem optimum. To this end, a selectorecombinative GA is used to search the minimum population size such that using random initialization it is able to converge to the optimum without any other mechanism than recombination and selection.

Table 3 depicts the method based on bisection. The method begins with a small population size which is doubled until the algorithm ensures a reliable convergence. We define the reliability criterion as the convergence of the algorithm to the optimum 49 out of 50 times (0.98 of SR). After that, the interval (min, max) is halved several times and the population size adjusted within such a range until $\frac{\max - \min}{\min} >$ threshold , where min and max stand respectively for the minimum and maximum population size estimated and threshold for the accuracy of the adjustment within such a range. This parameter has been set to $\frac{1}{16}$ in order to obtain a good adjustment of the population size.

Table 3 Population tuning algorithm based on bisection

```
N = Initial Population Size (20)
while GA reliability (N) < 98% do
  min = N; max‹ N = 2N
end while
while (max−min)/min > 1/16 do
  N = (max+min)/2
  if GA reliability(N) < 98% then
    min = N
  else
    max = N
  end if
end while
```

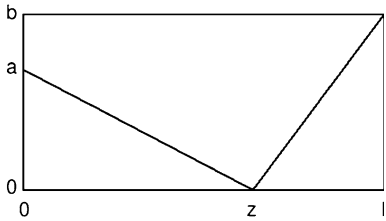

Fig. 7 Generalized *l-trap* function

4.2 The benchmark

Following Lobo and Lima's recommendations [28] about choosing a test suite with known population requirements and investigating the scalability on landscapes of different characteristics, experiments were conducted on trap functions [1]. A trap function is a piecewise-linear function defined on unitation (the number of ones in a binary string). There are two distinct regions in the search space, one leading to a global optimum and the other leading to the local optimum (see Fig. 7). In general, a trap function is defined by the following equation:

$$\text{trap}(u(\overrightarrow{x})) = \begin{cases} \frac{a}{z}(z - u(\overrightarrow{x})), & \text{if } u(\overrightarrow{x}) \leq z \\ \frac{b}{l-z}(u(\overrightarrow{x}) - z), & \text{otherwise} \end{cases} \quad (2)$$

where $u(\overrightarrow{x})$ is the unitation function, a is the local optimum, b is the global optimum, l is the problem size and z is a slope-change location separating the attraction basin of the two optima.

For the following experiments, 2-trap, 3-trap and 4-trap functions were designed with the following parameter values: $a = l - 1$, $b = l$, and $z = l - 1$. With these settings,[1] 2-trap is not deceptive, 4-trap is deceptive and 3-trap lies in the region between deception and non-deception. Under these conditions, it is possible not only to examine the scalability on trap functions, but also to investigate how the scalability varies when changing from non-deceptive to deceptive search landscapes. Scalability tests were performed by juxtaposing m trap functions in binary strings of length L and summing the fitness of each sub-function to obtain the total fitness.

All settings are summarized in Table 4,[2] operators as binary tournament or uniform crossover are standard in GAs.

The baseline for comparison are two panmictic GAs that follow an steady-state and a generational scheme and have no mutation in order to meet the selectorecombinative criterion of the bisection method. Since the methodology imposes a SR of 0.98 in the results, the $AESQ_3$ has been used as an appropriate metric to measure the computational effort to reach the success criterion. A more efficient algorithm needs a smaller number of evaluations. Finally, the cache size of newscast has been fixed to 20 based on the results in [26] which shows the robustness of the EvAg

[1] Originally, Ackley's trap functions use $z = \frac{3l}{4}$, however, [7] demonstrates that trap functions are fully easy under such settings.

[2] All the source code for the experiments is available from our Subversion repository at http://www.forja.rediris.es/svn/geneura/evogen published under GPL v3. Accessed on September 2009.

Table 4 Parameters of the experiments

Trap instances	
BB size	2, 3, 4
Individual length (L)	12, 24, 36, 48, 60
GA settings	
GA	Selectorecombinative SSGA
	Selectorecombinative GGA
	Selectorecombinative EvAg
Population size	Tuning algorithm
Selection of Parents	Binary Tournament
Recombination	Uniform crossover, $p_c = 1.0$
Newscast settings	
Cache size	20

model regarding such a parameter. Setting a different cache size does not alter the quality of the solutions.

5 Analysis of results

Figure 8 depicts the scalability of the population size (N) and the required number of evaluations to reach the problem optimum ($AESQ_3$) for the three approaches under study on 2, 3, and 4-trap functions. All the graphics show that either the population size or the computational effort fit with a polynomial order of scalability with base the length of the chromosome (L) and different exponents depending on the problem difficulty and the approach itself.

The first conclusion that can be easily drawn from results is a better scalability of the EvAg approach with respect to the population size, specially when the problem difficulty increases from 2 to 4-trap. That is, increasing the problem difficulty makes that the GGA and SSGA face extreme difficulties to track the problem optimum, thus requiring a higher population size N to prevent that the algorithm gets stuck in local optima. From a computational perspective, this fact can be translated into a more efficient use of the running platform since the EvAg approach will require a smaller amount of computational resources. Additionally, results on $AESQ_3$ are clearly correlated to the population size, SSGA scales better than GGA and it is roughly similar to EvAg in 2 and 3-trap. Nevertheless, as the problem difficulty increases to 4-trap, EvAg scales clearly better.

In the second series of experiments we consider a more realistic GA setup and switch mutation on. This implies that we have to specify values for the mutation rate parameter p_m. Strictly speaking, we should also recalibrate population sizes, since the bisection method only gives good estimates for selectorecombinative GAs. However, an extensive parameter sweep goes far beyond the scope of this paper and therefore we will use the common "$\frac{1}{L}$ heuristic" for setting the mutation rates and keep the population sizes that were used in the first series of experiments.

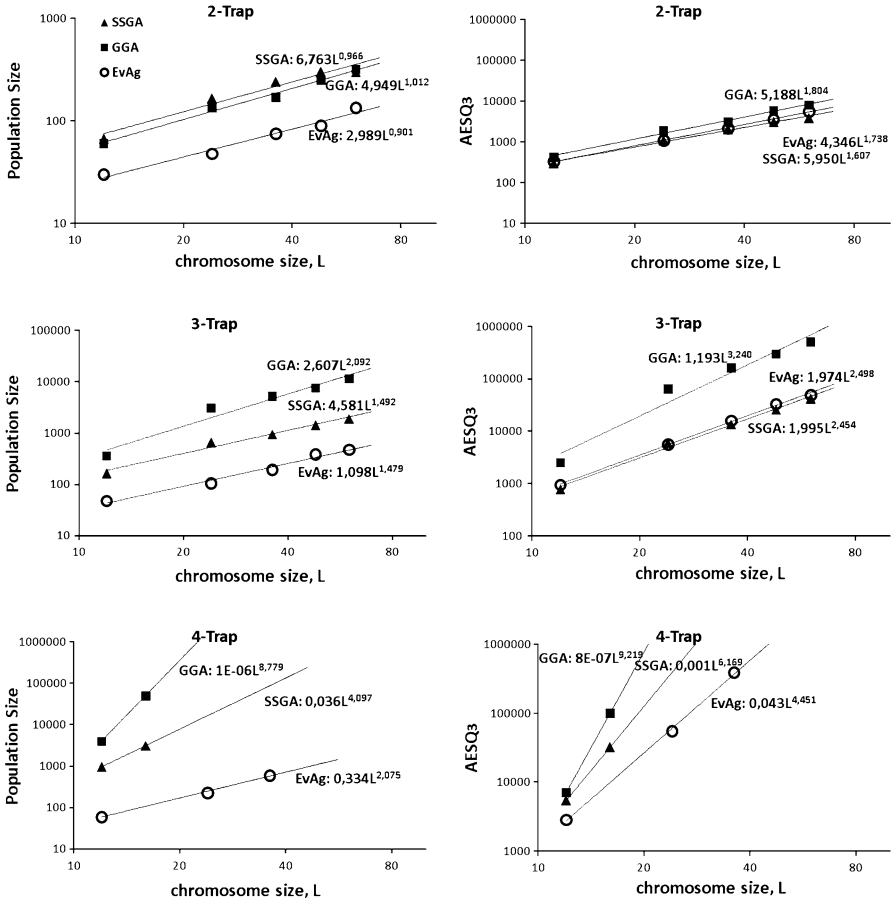

Fig. 8 Scalability in trap functions based on the population tuning algorithm and the selectorecombinative versions of the generational GA (*GGA*), steady-state GA (*SSGA*) and the Evolvable Agent (EvAg). On the left the estimated population sizes N and the evaluations to solution in third quartile $AESQ_3$ on the right. Results are obtained over 50 independent runs and depicted in a *log-log* scale as a function of the length of the chromosome, L

Obviously, in this case we only need to look at the $AESQ_3$ results. The outcomes of these experiments are shown in Fig. 9 in which curves appear shifted with respect to the selectorecombinative version in Fig. 8 but approximately keeping the same scalability order. Table 5 compares such estimated complexity orders with mutation switched off and on, showing that mutation does not alter the order in algorithm performance, and exponents are roughly the same, with only the constant in the power law changing.

In the last series of experiments we try to gain more detailed insights in the differences between the three approaches to population management. To this end, we run the GGA, SSGA, and EvAg models using the same settings on a problem instance, summarized in Table 6 (recall, that in the previous experiments, GGA,

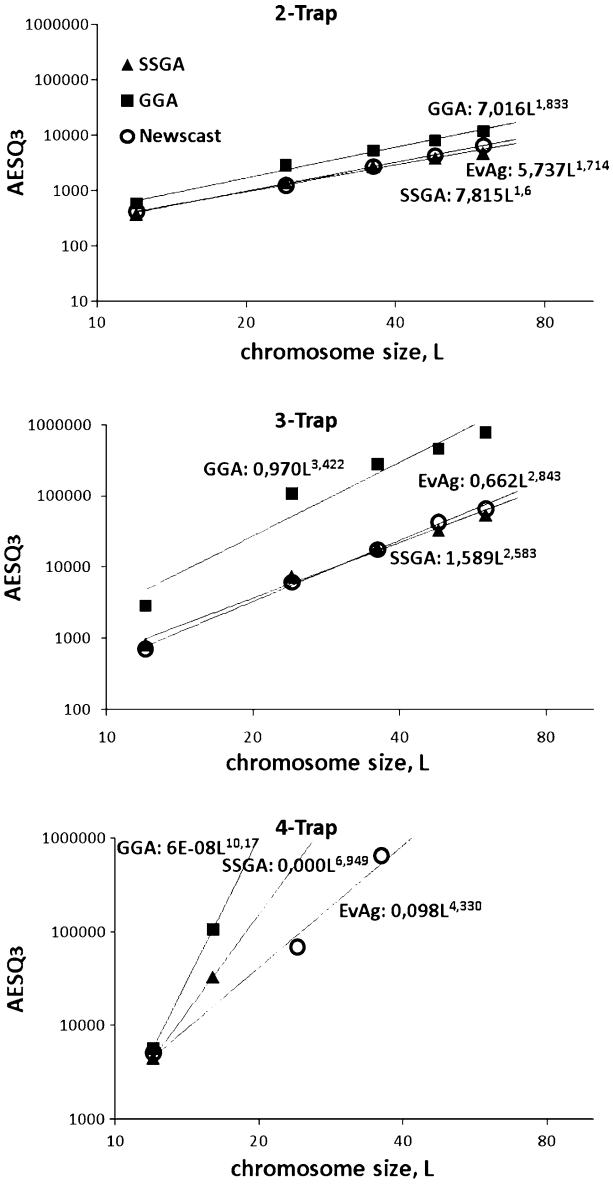

Fig. 9 Reproduction of the results in Fig. 8 with mutation switched on

SSGA, and EvAg used different parameters on any given problem instance). We choose the largest instance of the most difficult problem under study ($L = 36$ in 4-trap) for this purpose, where the differences in scalability between the approaches are most visible. On this instance, the population size for the selectorecombinative EvAg was estimated to 600 and the $AESQ_3$ to 393000. Switching mutation on implies that $N = 600$ is oversized for the EvAg approach since mutation represents

Table 5 Complexity orders of the $AESQ_3$ scalability in O notation

	GGA mutation		SSGA mutation		EvAg mutation	
	Off	On	Off	On	Off	On
2-Trap	$O(L^{1.804})$	$O(L^{1.833})$	$O(L^{1.607})$	$O(L^{1.6})$	$O(L^{1.738})$	$O(L^{1.714})$
3-Trap	$O(L^{3.24})$	$O(L^{3.422})$	$O(L^{2.454})$	$O(L^{2.583})$	$O(L^{2.498})$	$O(L^{2.843})$
4-Trap	$O(L^{9.219})$	$O(L^{10.17})$	$O(L^{6.169})$	$O(L^{6.949})$	$O(L^{4.451})$	$O(L^{4.33})$

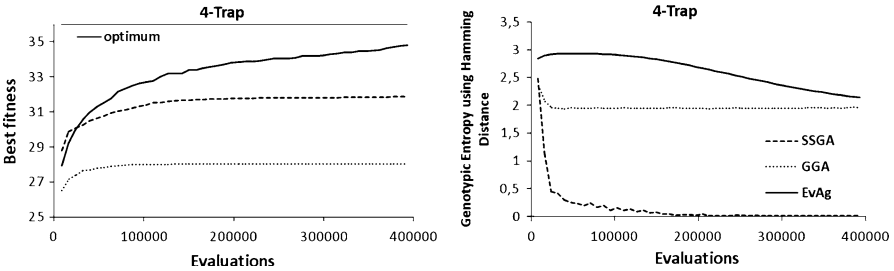

Fig. 10 Best fitness convergence (*left*) and evolution of the diversity expressed as the entropy based on the hamming distances between the genotypes (*right*). Graphs plotted represent the average of 50 independent runs

a new source of diversity, however, the highest scalability orders of SSGA and GGA indicate that such a population size will remain undersized in both cases.

The results in Fig. 10 show that EvAg is still converging towards the optimum while the SSGA and GGA stagnates at early stages of the search. Despite the SSGA performing a more exploitative search than the GGA, as shown by the genotypic entropy converging to zero, both cases lost track of the optimum, a fact that might be explained by an undersized population size. On the other hand, the evolution of diversity of the EvAg approach indicates that the algorithm is still converging when the termination criterion is met. Since a selectorecombinative EvAG is able to find the optimum in such number of evaluations it is straightforward to see that the population size is oversized in this case. Given that the three approaches are equally parameterized and that SSGA shares the same reproductive scheme than the EvAg, it is within the population structure of the EvAg where the genetic diversity is preserved at a higher level and consequently the population size N can be reduced. Hence, the EvAg approach scales better on difficult problems. With a lower optimal N, EvAg needs fewer evaluations to reach the optimum when compared to panmictic GAs.

Finally, the normality of such results were analyzed using the Anderson-Darling test which refutated the null hypothesis on the normality of the data. Therefore, a non-parametric Wilcoxon test was used to compare the quality of fitness between the EvAg and the SSGA and GGA approaches. Table 7 presents the Wilcoxon analysis of the data which shows significant differences for the EvAg approach with respect to the SSGA and the GGA.

Table 6 Parameters for the third series of experiments

Trap instance	
BB size	4
Individual length (L)	36
GA settings	
GA	SSGA
	GGA
	EvAg
Population size	600
Termination condition	Max. eval. = 393,000
Selection of parents	Binary tournament
Recombination	Uniform crossover, $p_c = 1.0$
Mutation	Bit-flip mutation, $p_m = \frac{1}{L}$
Newscast settings	
Cache size	20

Table 7 Wilcoxon test comparing the best fitness distributions of equally parameterized SSGA, GGA and EvAg in 4-trap

Problem instance	Algorithm	Avg. fitness $\pm$ σ	Wilcoxon test	Significantly different?
Trap4	GGA	28.02 $\pm$ 0.14	$W = 2{,}500$ p-value $< 2.22e\text{-}16$	Yes
$L = 36$	SSGA	31.88 $\pm$ 1.15	$W = 2{,}476$ p-value $< 2.22e\text{-}16$	Yes
$N = 600$				
M. Eval. = 393,000	EvAg	34.82 $\pm$ 0.66	–	–

Results are obtained over 50 independent runs

6 Conclusions

The Evolvable Agent is a spatially structured model designed for Evolutionary Computation in P2P infrastructures. The model defines the population structure by means of the gossiping protocol newscast that behaves asymptotically as a small-world graph. The influence of such kind of structures in the environmental selection pressure of EAs is close to that in panmictic populations used by default in canonical GA approaches. Nevertheless, as it has been shown in this paper the inhomogeneities of small-world structured population play an important role in the preservation of the genetic diversity, and have, therefore, a positive effect on scalability.

The EvAg model has demonstrated good scalability on trap functions with search landscapes of different difficulties. We found that EvAg scales better than a GGA and an SSGA that only differ from it in the population structure. Based on various scenarios with and without mutation we can conclude that EvAg needs fewer evaluations to reach a solution in addition to requiring smaller populations. The improvement is much more noticeable as the problem difficulty increases showing

thereby the adequacy of the P2P approach for tackling large instances of difficult problems.

To gain more detailed insights, the runtime behavior of equally parameterized EvAg, SSGA, and GGA approaches has been analyzed in depth on our largest fully deceptive problem instance. The results show that EvAg can maintain higher population diversity and better progress in fitness. As a consequence, an oversized population for the EvAg model still remains undersized for the canonical approaches that get lost in local optima.

As future lines of work, we intend to assess the EvAg behavior on a wider range of problem landscapes and study the dynamics of the approach taking into account issues such as asynchrony, heterogeneity and fault tolerance in addition to the study of churn in [23].

Acknowledgments This work has been supported by the Spanish MICYT project TIN2007-68083-C02-01, the Junta de Andalucia CICE project P06-TIC-02025 and the Granada University PIUGR 9/11/06 project.

References

1. D.H. Ackley, *A Connectionist Machine for Genetic Hillclimbing*. (Kluwer, Norwell, MA, 1987)
2. E. Alba, M. Tomassini, Parallelism and evolutionary algorithms. IEEE Trans. Evol. Comput. **6**(5), 443–462 (2002)
3. M. Arenas, P. Collet, A.E. Eiben, M. Jelasity, J.J. Merelo, B. Paechter, M. Preuss, M. Schoenauer, A framework for distributed evolutionary algorithms. In *Parallel Problem Solving from Nature—PPSN VII, Granada, Spain*, No. 2439 in Lecture Notes in Computer Science, LNCS. (Springer, 2002), pp. 665–675
4. B. Bánhelyi, M. Biazzini, A. Montresor, M. Jelasity, Peer-to-peer optimization in large unreliable networks with branch-and-bound and particle swarms. In *Applications of Evolutionary Computing*, Lecture Notes in Computer Science, ed. by M. Giacobini, A. Brabazon, S. Cagnoni, G.A.D. Caro, A. Ekárt, A.I. Esparcia-Alcázar, M. Farooq, A. Fink, P. Machado (Springer, 2009), pp. 87–92
5. J. Berntsson, G2DGA: an adaptive framework for internet-based distributed genetic algorithms. In *GECCO '05: Proceedings of the 2005 workshops on Genetic and evolutionary computation*, pp. 346–349
6. E. Cantú-Paz, *Efficient and Accurate Parallel Genetic Algorithms*. (Kluwer, Norwell, MA, 2000)
7. K. Deb, D.E. Goldberg, Analyzing deception in trap functions. In *Foundations of Genetic Algorithms*. (Morgan Kaufmann, 1991), pp. 93–108
8. A.E. Eiben, J.E. Smith, *Introduction to Evolutionary Computing*. (Springer, Berlin, 2003)
9. A.E. Eiben, A.R. Griffioen, E. Haasdijk, Population-based adaptive systems: concepts, issues, and the platform NEW TIES. In *Proceedings of European Conference on Complex Systems, Dresden, Germany*, http://www.cs.vu.nl/~gusz/papers/2007-ECCS-PAS.pdf
10. G. Folino, G. Spezzano, P-cage: an environment for evolutionary computation in peer-to-peer systems. In *EuroGP*, Lecture Notes in Computer Science, vol. 3905, ed. by P. Collet, M. Tomassini, M. Ebner, S. Gustafson, A. Ekárt (Springer, 2006), pp. 341–350
11. M. Giacobini, E. Alba, A. Tettamanzi, M. Tomassini, Modeling selection intensity for toroidal cellular evolutionary algorithms. In *GECCO '04: Proceedings of the 2004 conference on Genetic and Evolutionary Computation* (Springer, Berlin/Heidelberg, LNCS, 2004), pp. 1138–1149
12. M. Giacobini, M. Tomassini, A. Tettamanzi, Takeover time curves in random and small-world structured populations. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation* (ACM, New York, NY, 2005a), pp. 1333–1340. http://www.doi.acm.org/10.1145/1068009.1068224
13. M. Giacobini, M. Tomassini, A. Tettamanzi, E. Alba, Selection intensity in cellular evolutionary algorithms for regular lattices. IEEE Trans. Evol. Comput. **9**(5), 489–505 (2005b)

14. M. Giacobini, M. Preuss, M. Tomassini, Effects of scale-free and small-world topologies on binary coded self-adaptive CEA. In *Evolutionary Computation in Combinatorial Optimization—EvoCOP 2006*, vol. 3906, ed. by J. Gottlieb, G.R. Raidl (Springer, Budapest, LNCS, 2006), pp. 85–96
15. D.E. Goldberg, *The Design of Innovation—Lessons from and for Competent Genetic Algorithms* (Kluwer, Norwell, MA, 2002)
16. D.E. Goldberg, K. Deb, A comparative analysis of selection schemes used in genetic algorithms. In: *Foundations of Genetic Algorithms* (Morgan Kaufmann, 1991), pp. 69–93
17. G. Harik, E. Cantú-Paz, D. Goldberg, B. Miller, The Gambler's ruin problem, genetic algorithms, and the sizing of populations. Evol. Comput. **7**(3), 231–253 (1999)
18. I. Hidalgo, F. Fernández, Balancing the computation effort in genetic algorithms. In *The 2005 IEEE Congress on Evolutionary Computation*, vol. 2 (IEEE Press, 2005), pp. 1645–1652. doi:10.1109/CEC.2005.1554886
19. M. Jelasity, M. van Steen, *Large-scale Newscast Computing on the Internet*. Tech. Rep. IR-503 (Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands). http://www.cs.vu.nl/pub/papers/globe/IR-503.02.pdf
20. M. Jelasity, A. Montresor, O. Babaoglu, Gossip-based aggregation in large dynamic networks. ACM Trans. Comput. Syst. **23**(3), 219–252 (2005)
21. J.L.J. Laredo, P.A. Castillo, B. Paechter, A.M. Mora, E. Alfaro-Cid, A. Esparcia-Alcázar, J.J. Merelo, Empirical validation of a gossiping communication mechanism for parallel EAs. In *EvoWorkshops*, Lecture Notes in Computer Science, vol. 4448 (Springer, 2007), pp. 129–136
22. J.L.J. Laredo, P.A. Castillo, A.M. Mora, J.J. Merelo, Exploring population structures for locally concurrent and massively parallel evolutionary algorithms. In *IEEE Congress on Evolutionary Computation (CEC2008), WCCI2008 Proceedings* (IEEE Press, Hong Kong, 2008a), pp. 2610–2617
23. J.L.J. Laredo, P.A. Castillo, A.M. Mora, J.J. Merelo, C. Fernandes, Resilience to churn of a peer-to-peer evolutionary algorithm. Int. J. High Perform. Syst. Archit. **1**(4), 260–268 (2008b). http://www.dx.doi.org/10.1504/IJHPSA.2008.024210
24. J.L.J. Laredo, A.E. Eiben, M. van Steen, J.J. Merelo, On the run-time dynamics of a peer-to-peer evolutionary algorithm. In *Proceedings of the 10th international conference on Parallel Problem Solving from Nature* (Springer, Berlin, Heidelberg, 2008c), pp. 236–245. http://www.dx.doi.org/10.1007/978-3-540-87700-4_24
25. J.L.J. Laredo, P.A. Castillo, A.M. Mora, J.J. Merelo, Evolvable agents, a fine grained approach for distributed evolutionary computing: walking towards the peer-to-peer computing frontiers. Soft Comput. Fusion Found. Methodol. Appl. **12**(12), 1145–1156 (2008d)
26. J.L.J. Laredo, C. Fernandes, A. Mora, P.A. Castillo, P. Garcia-Sanchez, J.J. Merelo, Studying the Cache Size in a Gossip-based Evolutionary Algorithm. In *3rd International Symposium on Intelligent Distributed Computing* (Springer, Berlin/Heidelberg, 2009), pp. 131–140
27. W.P. Lee, Parallelizing evolutionary computation: a mobile agent-based approach. Expert Syst. Appl. **32**(2), 318–328 (2007)
28. F.G. Lobo, C.F. Lima, Adaptive population sizing schemes in genetic algorithms. In *Parameter Setting in Evolutionary Algorithms, Studies in Computational Intelligence* (Springer, Berlin/Heidelberg, 2007), pp. 185–204
29. J.J. Merelo, P.A. Castillo, J.L.J. Laredo, A.M. Mora, A. Prieto, Asynchronous distributed genetic algorithms with Javascript and JSON. In *IEEE Congress on Evolutionary Computation (CEC2008), WCCI2008 Proceedings* (IEEE Press, Hong Kong, 2008), pp. 1372–1379
30. N. Nedjah, L. de Macedo Mourelle, E. Alba (eds.), *Parallel Evolutionary Computations, Studies in Computational Intelligence*, vol. 22. (Springer, 2006)
31. M. Preuss, C. Lasarczyk, On the importance of information speed in structured populations. In *PPSN*, vol. 3242 (Springer, 2004), pp. 91–100
32. K. Sastry, *Evaluation-relaxation Schemes for Genetic and Evolutionary Algorithms*. Tech. Rep. 2002004 (University of Illinois at Urbana-Champaign, Urbana, IL, 2001)
33. R. Steinmetz, K. Wehrle, What is this peer-to-peer about? In *Peer-to-Peer Systems and Applications*, Lecture Notes in Computer Science, vol. 3485, ed. by R. Steinmetz, K. Wehrle (Springer, 2005), pp 9–16
34. D. Thierens, Scalability problems of simple genetic algorithms. Evol. Comput. **7**(4), 331–352 (2005)
35. M. Tomassini, *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time (Natural Computing Series)* (Springer New York, Inc., Secaucus, NJ, 2005)

36. S. Voulgaris, M. Jelasity, M. van Steen, *A Robust and Scalable Peer-to-Peer Gossiping Protocol*, Lecture Notes in Computer Science (LNCS), vol. 2872 (Springer, Berlin/Heidelberg, 2004), pp. 47–58. doi:10.1007/b104265
37. D. Watts, S. Strogatz, Collective dynamics of "small-world" networks. Nature **393**, 440–442 (1998). http://www.dx.doi.org/10.1038/30918
38. J. Whitacre, R. Sarker, Q. Pham, The self-organization of interaction networks for nature-inspired optimization. IEEE Trans. Evol. Comput. **12**(2), 220–230 (2008). doi:10.1109/TEVC.2007.900327
39. W.R.M.U.K. Wickramasinghe, M. van Steen, A.E. Eiben, Peer-to-peer evolutionary algorithms with adaptive autonomous selection. In *GECCO '07* (ACM Press, New York, NY, 2007), pp. 1460–1467, http://www.doi.acm.org/10.1145/1276958.1277225